Planung und Umsetzung einer Medieninstallation mit dem Schwerpunkt: Steuerung über DMX und RS232

Bachelorarbeit

Vorgelegt im Studiengang Medien und Informationswesen
der Hochschule Offenburg

Erstbetreuer: Prof. Dr. Dan Curticapean
Zweitbetreuer: Benjamin Heitz M. Sc.

Marina Zajec
Matrikelnummer: 174057
Theobald-Kernerstraße 9
70372 Stuttgart
marina.zajec@web.de

Offenburg, den 16.07.2015

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ANSI	American National Standards Institute
CMY	Cyan, Magenta, Yellow
COM	Communication Equipment
DIP	Dual in-line package
DMX	Digital Multiplex
km	Kilometer
LED	light- emitting diode
m	Meter
RGB	Rot, Grün, Blau
SD	Secure Digital
SMT	Simple Multi- Touch
XLR	Screen (X), Life (L), Return (R)

1. Einleitung

Das Museum im Ritterhaus in Offenburg ist ein Museum für alle Altersklassen. Es beschäftigt sich mit archäologischen Ausgrabungsstücken, religiöser Volkskunst, regionaler Naturkunde, Geologie und kolonialzeitlicher Völkerkunde.

Durch einen Umbau und innovative Einrichtungen will das Museum das Interesse und die Neugierde an Natur und Geologie wecken. Dieses Vorhaben wird mit der Installation einer Zeitmaschine umgesetzt. Im Dachgeschoss des Museums wird der Besucher in einer Zeitkapsel in Empfang genommen bevor es zu der Erkundung der Stollen weiter geht. Durch die Zeitmaschine sollen Kinder aus der gewohnten Museumsumgebung in eine andere Welt eintauchen. Der hierbei verwendete Einsatz von Multimedia und Interaktion ermöglicht es dem Museum den Besuchern die verschiedenen Erdzeitalter auf eine spielerische Art näher zu bringen.

Zudem wird der Lern- und Spaßerfolg zum einen durch die Anregung der unterschiedlichen menschlichen Sinne Sehen, Hören und Tasten unterstützt, zum anderen haben die Besucher die Möglichkeit den Inhalt beliebig oft zu wiederholen.

Damit die einzelnen Elemente der Zeitkapsel für den Besucher ein zusammengehöriges und in sich stimmiges Erlebnis ergeben, ist eine technische Planung notwendig. Die zentrale Herausforderung ist die Verknüpfung und Ansteuerung der einzelnen Komponenten über eine Software. Diese dient dem Besucher zudem als Benutzeroberfläche und ermöglicht das beliebige Steuern und Interagieren.

Die vorliegende Arbeit stellt zunächst das Konzept und die Idee der Installation vor. Anschließend werden die eingesetzte Technik und das verwendete Protokoll erklärt. Daraufhin nennt die Autorin die einzelnen Bauelemente und Geräte, erklärt wie diese für die Zeitmaschine zum Einsatz gebracht werden und stellt die technische Planung vor. Abschließend beschäftigt sich die Ausarbeitung mit der Software und dem Programmcode zur Steuerung der einzelnen Elemente.

2. Zeitmaschine

Die Zeitmaschine soll vor allem Kindern einen Einstieg in das Thema Geologie ermöglichen. Es ist vorgesehen die verschiedenen Erdzeitalter zu erkunden, in denen die unterschiedlichen Gesteinsarten entstanden sind.

Wie man im folgenden Bild sehen kann, ist die Zeitmaschine als Kapsel aus fünf- und sechseckigen Holzplatten gestaltet (Abbildung 1).

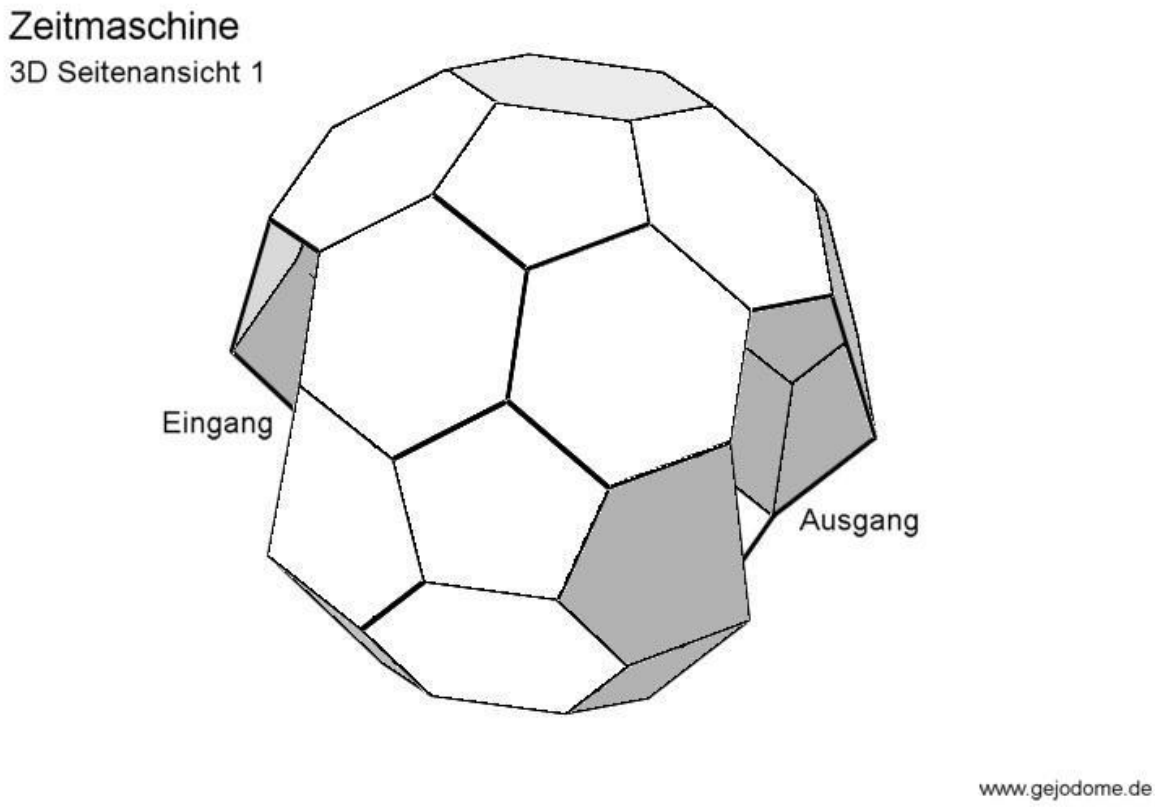


Abbildung 1: Kuppel in 3D Ansicht (Home - GEjODOME Zelte - der Dome aus Holz)

In der Kapsel kann man über den Touchbildschirm eines der für diese Region relevanten Erdzeitalter auswählen: Ordovizium, Karbon, Perm, Trias, Jura, Tertiär, Eiszeit und Heute.

Durch die Auswahl werden mehrere Effekte ausgelöst: Der Besucher sieht auf dem Touchbildschirm eine Animation der Kontinente um deren Position im ausgewählten Zeitalter zu sehen. Eine Stimme kündigt die Ankunft im jeweiligen Erdzeitalter an und eine digital animierte Landschaft ist auf einem weiteren Bildschirm an der Wandinnenseite zu sehen. Zudem werden die jeweils in dem Zeitalter vorkommenden Steine und Fossilien, welche ebenfalls an der Kuppelinnenseite angebracht sind, beleuchtet.

Der Ablauf der verschiedenen Effekte sieht folgendermaßen aus:

Der Benutzer wählt ein Zeitalter, daraufhin startet die Animation mit einem Startgeräusch und während dem Ablauf der Kontinentaldrift sind Reisegeräusche zu hören. Bei Ankunft im ausgewählten Erdzeitalter ertönt ein Ankunftston und auf dem anderen Bildschirm beginnt ein Landschaftsvideo, welches mit einer informativen Sprachaufnahme unterlegt ist.

Um diese Effekte bei Berührung des jeweiligen Erdzeitalters auszulösen, bedient man sich verschiedener Techniken um Signale an externe Geräte zu senden und diese somit zu steuern. Die theoretischen Grundlagen zu dieser Thematik folgen in den nächsten Kapiteln.

3. DMX

3.1 Von analog zu digital

Früher wurden Steuersignale über analoge Gleichspannungen übertragen. Bei dieser Methode benötigt jeder Kanal seine eigene Ader. Das führt neben den Kosten, dem logistischen Aufwand und dem Transportaufwand auch zu Schwierigkeiten bei der Reparatur vor Ort. Hinzu kommen zu steuernde bewegte Scheinwerfer, welche mehrere Kanäle voraussetzen und über die analoge Datenübertragung nicht mehr realisierbar sind (vgl. Bewer und Steckmann 2004, S. 217–218).

Heutzutage werden diese Signale auf digitalem Wege übertragen, da durch die „Notwendigkeit, immer mehr Regelkreise zu bedienen und immer exaktere Steuerungen vorzunehmen, […] der Bedarf für leistungsfähigere Systeme, die nur auf digitaler Basis zufrieden stellend funktionieren können, [zunahm, M.Z.]“ (Bewer und Steckmann 2004, S. 118).

3.2 *DMX 512* Standard

Der *DMX 512* Standard wurde durch das amerikanische *USITT (United States Institute of Theatre Technology, Inc.)* dokumentiert und publiziert. Er entwickelte sich im Jahr 1986, wurde 1990 überarbeitet und von dem *IEC Technical Committee 34* zum *Standard Entertainment Technology - USITT DMX512-A (IEC 62136)* erklärt.
Inzwischen ist *DMX 512* auch ein *ANSI Standard* und somit international bekannt (vgl. USITT).

Das *USITT* beschreibt diese technische Entwicklung folgendermaßen: „This standard is intended to provide for interoperability at both communication and mechanical levels with controllers made by different manufacturers“ (vgl. USITT).

3.3 *DMX 512*-Steuerung

Die *DMX 512*-Steuerung ist ein digitales Schnittstellenprotokoll und ermöglicht die Steuerung und Kontrolle von Scheinwerfern, Dimmern, LED-Strahlern, Moving Heads und weiteren Geräten mithilfe von Kanälen.
Über DMX kann man zahlreiche Eigenschaften von Geräten steuern. Im Folgenden werden einige dieser Eigenschaften aufgezählt:

Helligkeit, Farbe, RGB/CMY, Pan (Schwenk), Tilt (Neigung), Gobo (Metall- oder Glasscheiben mit eingestanztem Muster um dieses auf die Wand zu projizieren), Shutter/Stroboskop (schnelles Öffnen/Schließen der Lichtöffnung), Geschwindigkeit und Fokus (vgl. Burghardt 2009, S. 26).

Neben der leitungsgebundenen Version gibt es auch *Wireless DMX*, was noch beweglichere Effekte ermöglicht. Zudem können über DMX-fähige Medienserver Video- und Tondateien gesteuert werden (vgl. Nöding 2015).

3.4 Verbindung und Anschluss von DMX-fähigen Geräten

Alle Geräte, welche über DMX angesteuert werden, sind in Reihe geschalten, um so den Datenstrom von Gerät zu Gerät durch zu leiten (Abbildung 2).
Dieses serielle Prinzip wird auch *Daisy Chain* genannt (vgl. Daisy-Chaining : daisy chaining : ITWissen.info 2004).

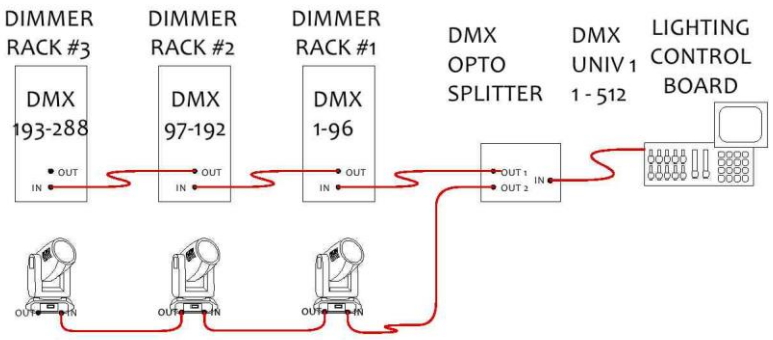


Abbildung 2: DMX Kette (designandtechtheatre on WordPress.com)

Um das Durchleiten zu garantieren sind überall zwei XLR-Buchsen angebracht. Diese dienen zum einen als Eingang und zum anderen als Durchschleif-Ausgang (vgl. Bewer und Steckmann 2004, S. 119–120).

Auf der folgenden Abbildung ist der Eingang als männliche Kupplung und der Ausgang als weibliche Kupplung sichtbar (Abbildung 3).

Abbildung 3: Output und Input zum Durchschleifen (13db Audio Education)

Nach dem *RS-485 Standard* sollen DMX-Kabel zwei-polig verdrillt und abgeschirmt sein, was auch als *shielded twisted-pair- Kabel* bezeichnet wird (vgl. Burghardt 2009, S. 34).

Wie man in der Abbildung sehen kann werden jeweils zwei Adernpaare miteinander verdrillt (Abbildung 4). Durch die Verdrillung vermindert sich der Einfluss von äußeren magnetischen oder elektrostatischen Feldern und das Übersprechen von Signalen. Zudem dient die Abschirmung der Reduktion von Interferenzen (vgl. Halsall 1996, S. 26).

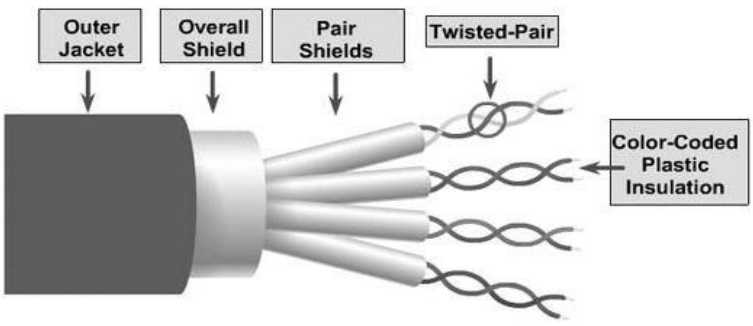


Abbildung 4: „shielded twisted-pair" Kabel (Lambda Group : Innovative Solutions for Infrastructure | Telecom Industry | Defence | Satellite Industry.)

Des Weiteren schreibt der *RS-485 Standard* eine maximale Kabellänge von einem Kilometer vor. In der Praxis ist diese Strecke jedoch nicht umsetzbar. Fachexperten raten zu einer Länge von etwa 500 m (vgl. Box 2010).

Die DMX-Norm gibt fünfpolige XLR-Stecker und Buchsen vor. Wie man im nachfolgenden Schaubild sehen kann, sind von diesen zum jetzigen Zeitpunkt nur drei belegt. Auf Pin 1 liegt die Masse beziehungsweise Abschirmung, auf Pin 2 das gegenphasige Signal und auf Pin 3 das phasenrichtige Signal (vgl. Ebner 2011, S. 115) (Abbildung 5). Diese Belegung bringt die Möglichkeit mit sich anstatt eines DMX-Kabels ein XLR-Kabel zu verwenden. Dabei sollten jedoch die schlechteren Übertragungseigenschaften des XLR-Kabels bedacht werden (vgl. Bewer und Steckmann 2004, S. 120).

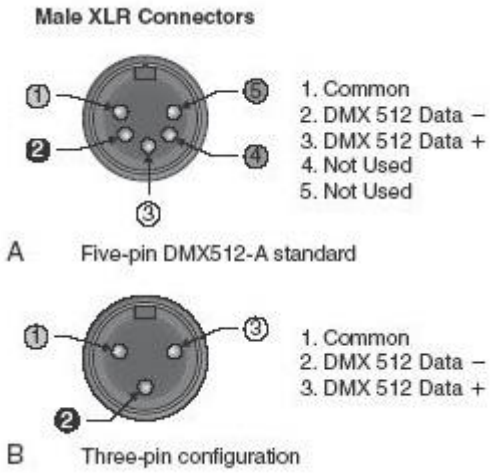


Abbildung 5: Pin Belegung eines DMX Steckerverbinders (Box 2010)

Bis zum heutigen Stand bietet DMX keinen Rückkanal vom Empfänger zum Sender. Dadurch ist auch keine Fehlerkorrektur möglich. Allerdings haben einzelne fehlerhafte Bytes keine schwerwiegenden Auswirkungen, da sie im darauffolgenden Datenpaket nochmals fehlerfrei gesendet werden können (vgl. Burghardt 2009, S. 27).

3.5 Steuerung und Adressierung

Es existieren zwei grundlegende Steuerungsmöglichkeiten von DMX Signalen. Zum einen gibt es den Weg über eine Hardwarelösung durch ein Pult und zum anderen über eine PC-basierte Software Lösung. Je nach Verwendungszweck wird zwischen diesen beiden Methoden entschieden (vgl. Burghardt 2009, S. 15).

Durch die Verwendung einer dieser Möglichkeiten und DMX können 512 verschiedene Kanäle angesteuert werden. Diesen Block von „512 Steuerungskanälen, die von einem Sender bedient werden“ (Burghardt 2009, S. 24) nennt man DMX-Universum. Hierfür wird jedem Gerät durch die Adressierung eine dreistellige Startadresse zwischen 001 und 512 zugewiesen. Diese kann man über verschiedene Vorrichtungen am jeweiligen Gerät einstellen (vgl. Box 2010).

Es gibt die Möglichkeit die Adresse über ein Interface, über Drehschalter oder über einen DIP- Schalter einzustellen (Abbildung 6 und Abbildung 7). DIP- Schalter sind kleine Schalterreihen und dienen zur Konfiguration von Geräten. Sie bestehen aus mehreren Einzelschaltern und können je nach Funktion ein- und ausgeschaltet werden (vgl. professional).

Abbildung 6: DMX Adressierung über ein Interface (Cameo Light)

Abbildung 7: DMX Adressierung über einen DIP- Schalter (Cameo Light)

Über die einzelnen Kanäle ist es dann möglich, individuelle Level- beziehungsweise Positionseinstellungen an die verschiedenen Geräte zu senden. Diese wiederum können die Daten aus dem Datenstrom herauslesen (vgl. Bewer und Steckmann 2004, S. 121).
Die 512 Kanäle haben eine Auflösung von 8 Bit und können somit jeweils einen Wert in der Spanne von 0 bis 256 erhalten (vgl. Ebner 2011, S. 117).
Dabei ist es auch möglich mehreren Geräten dieselbe Startadresse zu vergeben. Das hat den Effekt, dass die Geräte bei Ansteuerung dieselben Eigenschaften und dasselbe Verhalten annehmen.

Zur Veranschaulichung folgt nun ein Beispiel:

Man hat zwei vierkanalige Dimmerpacks. Das zweite soll nun über die DMX-Kanäle 5 bis 9 angesteuert werden. Somit erhält dieser Dimmer die Startadresse 5 und reagiert ab 5 aufwärts.

Gerät	**Kanäle**	**Belegter Adressbereich**
Dimmerpack 1	4	1-4
Dimmerpack 2	4	5-9

Tabelle 1: Adressierung mehrerer Geräte (Eigene Darstellung)

Bei Dimmern entspricht der zu übertragende Wert 0 einer Intensität von 0% und 255 einer Intensität von 100% (vgl. Schiller 2010, S. 7).

4. Geräte

Für die Umsetzung der Zeitmaschine werden verschiedene Geräte verbaut, welche ebenfalls in Anhang B in der Geräteliste vorzufinden sind. Im Folgenden werden all diese Geräte erst allgemein vorgestellt und im Anschluss wird der jeweils ausgewählte Gerätetyp näher beschrieben.

Zudem wird aufgeführt welche Funktion sie in der Zeitmaschine übernehmen.

4.1 Dimmerpack

„Ein Dimmerpack dimmt die Lampen. Es besteht also die Möglichkeit die Helligkeit einer Lampe stufenlos einzustellen“ (Burghardt 2009, S. 44). Dabei ist es möglich die Helligkeitswerte zwischen 0% und 100% einzustellen (vgl. Bewer und Steckmann 2004, S. 123).

Abbildung 8: Dimmerpack (Musikinstrumente bei session)

Dimmer funktionieren nach dem Prinzip der Phasenanschnittssteuerung. Nach einer Definition von Lindner, Brauer und Lehmann wird dem Verbraucher hierbei „nur während eines Teils der Periodendauer der Wechselspannung ein Strom geliefert“ (Lindner et al. 2004, S. 301).

Der Stromfluss im Lastkreis wird je Periode nur für ein bestimmtes Zeitintervall freigegeben. Dieses Zeitintervall kann man durch einen elektrischen Schalter beliebig anpassen.
In einem Dimmer ist entweder das elektrische Bauelement Triac oder ein Thyristor verbaut. Zunächst ist der Schalter in Durchlassrichtung geöffnet und somit kann auch kein Strom durchfließen (vgl. Müller 2006, S. 89).

Sobald die Verzögerungszeit abläuft und das Gate mit einem Stromimpuls angesteuert wird, wird der Thyristor leitend. Hier sieht man den Thyristor in zündendem Zustand (Abbildung 9).

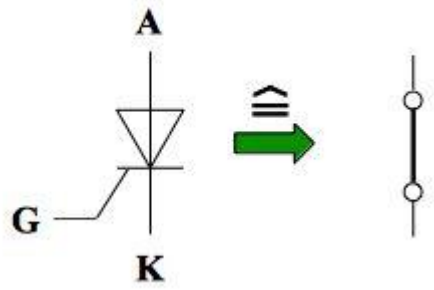


Abbildung 9: Thyristor im gezündeten Zustand (Thyristor Grundlagen)

Dieser bleibt solange leitend, bis er wieder Null ist. Wie man in folgender Abbildung sehen kann wartet er von diesem Zeitpunkt an auf den nächsten Zündimpuls, auch Gatestromimpuls genannt (vgl. Kories und Schmidt-Walter 2010, S. 572) (Abbildung 10).

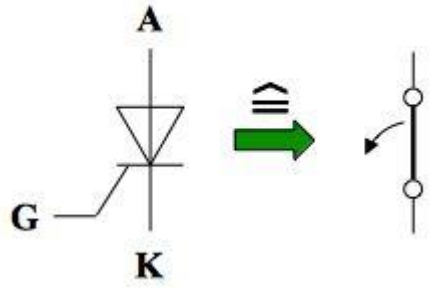


Abbildung 10: Thyristor im wiederherstellenden Zustand (Thyristor Grundlagen)

Durch das schnelle Schalten vom Sperrzustand in den Durchlasszustand ist dieses elektrische Bauelement für einen Dimmer gut geeignet.

Anstatt eines Thyristors kann auch ein Triac verwendet werden. Dieser besteht aus zwei antiparallel geschalteten Thyristoren und wird ebenfalls durch einen Stromimpuls leitend, hat jedoch keine bestimmte Durchlassrichtung (vgl. Müller 2006, S. 470–471).

Anhand folgender Abbildung wird das Prinzip eines Dimmers nochmals aufgezeigt: Nur ein Teil der Wechselspannung wird genutzt und kann am Verbraucher durch die Änderung des Mittelwertes der Spannung verändert werden (Abbildung 11).

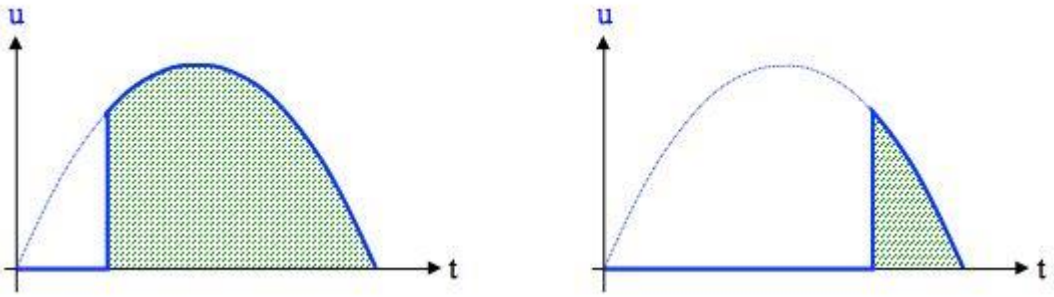


Abbildung 11: verschiedene Mittelwerte durch eine Phasenanschnittssteuerung (Thyristor Grundlagen)

Für den Fall, dass die Dimmerkurve einstellbar ist und somit nicht zwingend linear sein muss, kann zwischen quadratisch, Vorheizung und Switch unterschieden werden (Abbildung 12). Durch die Dimmerkurve kann man den Unterschied zwischen dem Eingangssignal und dem daraus resultierenden Ergebnis einstellen (vgl. Burghardt 2009, S. 45).

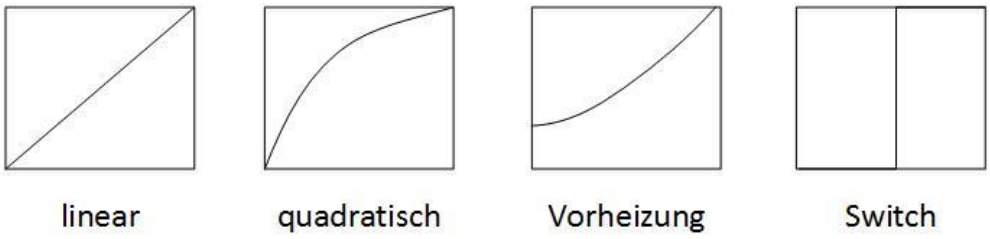


Abbildung 12: Dimmerkurven (Eigene Darstellung, angelehnt an: Burghardt 2009)

4.1.1 EDX-4 von Eurolite

EDX-4 von dem Hersteller *Eurolite* ist ein 4-Kanal-Dimmer und verfügt über vier Schutzkontaktsteckdosen als Ausgang. Diese werden an den Endverbrauchern, in dieser Installation an die Lampen angeschlossen.

Damit die Signale an dem jeweiligen Gerät ankommen, muss die Startadresse über das Interface eingestellt werden.

4.2 Switchpack

Im Gegensatz zu Dimmern sind Switchpacks reine Schaltvorrichtungen. „Ein Switchpack ist wie ein Schalter, er kennt nur die Zustände *AN* und *AUS*. Ab einem bestimmten DMX- Wert (z.B. 50% bzw. DMX- Wert 128) schaltet das Pack durch und die Lampe brennt“ (Burghardt 2009, S. 44).

Abbildung 13: Switchpack (Conrad)

Der Nachteil davon ist, dass die Lebensdauer von dem Leuchtmittel durch das ständige An- und Ausschalten reduziert wird (vgl. Burghardt 2009, S. 44).

4.2.1 ERX-4 von Eurolite

Das 4-Kanal-Switchpack *ERX-4* von *Eurolite* ist wie das Dimmerpack aufgebaut, wie schon im Kapitel „4.1.1 EDX-4 von Eurolite“ beschrieben und verfügt über die gleiche technische Spezifikation. Die Adressierung erfolgt allerdings nicht über ein Interface, sondern über neun DIP- Schalter.

Diese sind binär, können somit nur auf 1 oder 0 eingestellt werden.
Jeder dieser Schalter hat den Wert 2^n, wobei n für die Nummer des DIP- Schalters steht.

In folgender Tabelle kann man sehen welchen DMX Wert der jeweilige Schalter hat.

Schalter	1	2	3	4	5	6	7	8	9
Berechnung	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8
DMX Wert	1	2	4	8	16	32	64	128	256

Tabelle 2: DIP- Schalter Berechnung (Eigene Darstellung)

Wenn man den DIP Schalter 3 nun auf 1 und die restlichen auf 0 einstellt, wird die DMX Adresse 4 angesprochen. Je nachdem wie viele Kanäle das Gerät belegt, werden dann die nachfolgenden DMX Kanäle adressiert. Belegt das Gerät beispielsweise 4 Kanäle, werden die Kanäle 4, 5, 6 und 7 adressiert.

4.3 USB Interface

Da der Computer nicht fähig ist mit DMX- oder RS232- fähigen Geräten wie in dieser Installation dem Dimmer und Switcher beziehungsweise dem Videoplayer zu kommunizieren, ist es notwendig eine weitere Schnittstelle über ein Interface hinzuzufügen.

Abbildung 14: DMX USB Interface (Conrad)

Über das DMX USB Interface werden die Signale für die Lichtsteuerung gesendet und über das RS232 USB Interface die Signale für die Video- und Audiosteuerung (Abbildung 14 und Abbildung 15).

Ein DMX USB Interface „kann mit einer Vielzahl unterschiedlicher DMX Software-Lösungen betrieben werden. Es verfügt über einen eigenen Mikroprozessor, der das DMX-Signal erzeugt“ (Enttec - DMX USB Pro Interface : Präsentationstechnik).

Dadurch ist es dem Anwender möglich die durch die Software produzierten Befehle als DMX-Signale über das Interface an die Geräte weiterzuleiten.

Dasselbe Verfahren gilt auch für ein RS232 USB Interface.

Abbildung 15: RS232 USB Interface (Re-In Retail International GmbH)

4.3.1 DMX-USB Pro Mk2 und RS232-USB Manhattan

Das hier verwendete Modell *DMX-USB Pro Mk2* ist vom Hersteller *Enttec* und das RS232-USB Interface von Manhattan.
Damit die Interfaces Signale über die Software erhalten können, wird die COM Schnittstelle angesprochen. Diese Schnittstelle wird unter Windows über den Gerätemanager ermittelt und kann im Programmiercode weiter verwendet werden.

4.4 Videoplayer

Auf dem Videoplayer *MPL031* von *tbm* werden über eine SD- Karte Bild- und Tondateien gespeichert. Der Videoplayer ist RS232-fähig und kann somit über eingehende Signale angesteuert werden (Abbildung 16).

Abbildung 16: Videoplayer (Unternehmen | tbm GmbH)

4.5 Touchscreen

4.5.1 Touchscreen als Bildschirmoberfläche

Ein Touchscreen ist ein direktes Eingabegerät (Abbildung 17). Das Display ist berührungsempfindlich und kann die Position des Fingers ermitteln. Hierbei werden keine zusätzlichen Geräte wie eine Tastatur oder Maus zum Bedienen und Interagieren benötigt (vgl. Schenk und Rigoll 2010, S. 17).

Durch das Berühren der Bildschirmoberfläche löst der Nutzer verschiedene Aktionen aus.

Das einfache und intuitive Bedienen eignet sich besonders zur einfachen Informationsbeschaffung und zum schnellen Lernen. Vor allem die freien Gestaltungsmöglichkeiten lassen die Oberfläche für Kinder attraktiv und einladend wirken (vgl. Zühlke 2012, S. 212).

Abbildung 17: All-In-One Touchscreen (US)

4.5.2 Technologie

Es gibt verschiedene Technologien um aus der Berührung der Bildschirmoberfläche Koordinatenpaare zu bilden damit das System entsprechend auf die Interaktion mit dem Nutzer reagieren kann. Dies geschieht über Sensoren, welche Signale an die Elektronik senden.

In dieser Arbeit wird nur die resistive Technologie vorgestellt, da der verwendete Touchscreen darauf basiert.

Ein resistiver Touchscreen besteht aus zwei leitfähigen Schichten und einer sich dazwischen befindlichen isolierenden Schicht (Abbildung 18).
Sobald ein Druckkontakt auf die Bildschirmoberfläche ausgeübt wird, erzeugt die isolierende Schicht einen elektrischen Kontakt zwischen den leitenden Schichten.
Der dabei entstehende elektrische Widerstand ist von der Position des Druckpunktes abhängig. Der Widerstand teilt sich je nach Entfernung vom Druckpunkt zur leitenden Schicht auf diese auf. Über einen Spannungsteiler kann das Verhältnis der Widerstände und somit die Position der Druckpunkte ermittelt werden (vgl. Zühlke 2012, S. 215).

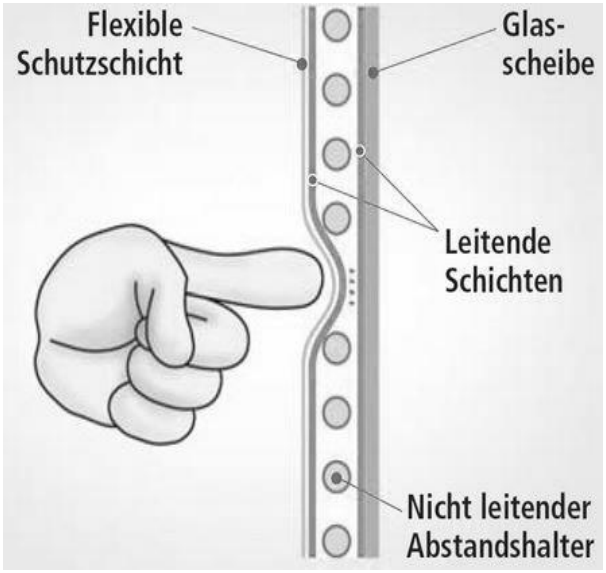


Abbildung 18: Prinzip der resistiven Bildschirmoberfläche (professional)

4.5.3 Das Userinterface für Kinder

Im Jahr 2006 führte die *Hochschule der Medien in Stuttgart* und die *User Interface Design GmbH (UID)* eine Studie im Bereich des *Usability Engineering für Kinder* durch. Der Kernbereich der Forschung bezieht sich auf eine effiziente und effektive Gestaltung der Oberfläche für Kinder.

Unter Berücksichtigung der Ergebnisse sind folgende Punkte für die Zielgruppe der Zeitmaschine relevant:

Die Aufmerksamkeitsspanne ist relativ gering und die Reaktionszeit etwas länger im Vergleich zu Erwachsenen. Auf dem Touchbildschirm werden deshalb anstatt langen Textpassagen nur kurze Begriffe aufgeführt.

Da die Feinmotorik noch nicht vollständig ausgebaut ist, sollten die Schaltflächen möglichst groß sein. Diese werden auf der rechten Seite des Bildschirms positioniert um das Verdecken anderer Elemente zu verhindern. Die Buttons haben zudem einen angemessen Abstand zueinander um der fehlenden Feinmotorik entgegen zu wirken.

Um dem Benutzer Rückmeldung auf seine Aktionen zu geben, können die Schaltflächen verschiedene Zustände einnehmen (vgl. Burmester et al., S. 4–26). Die Buttons sind zunächst in dem Zustand *inaktiv* (Abbildung 19). Sobald ein Erdzeitalter ausgewählt ist, fällt er in den Zustand *aktiv* und färbt sich in einen anderen Farbton ein (Abbildung 20).

Abbildung 19: Button Ordovizium im Zustand inaktiv (Fofana 2015)

Abbildung 20: Button Ordovizium im Zustand aktiv (Fofana 2015)

Jedem Erdzeitalter wird für den aktiven Zustand eine andere Farbe zugewiesen, welche auch bei dem Zeitstrahl verwendet wird. Die Anfertigung der Benutzeroberfläche durch eine Grafikerin verbindet die einzelnen Elemente Schaltflächen, Zeitstrahl, Timer und Titel zu einem Gesamtbild (Abbildung 21).

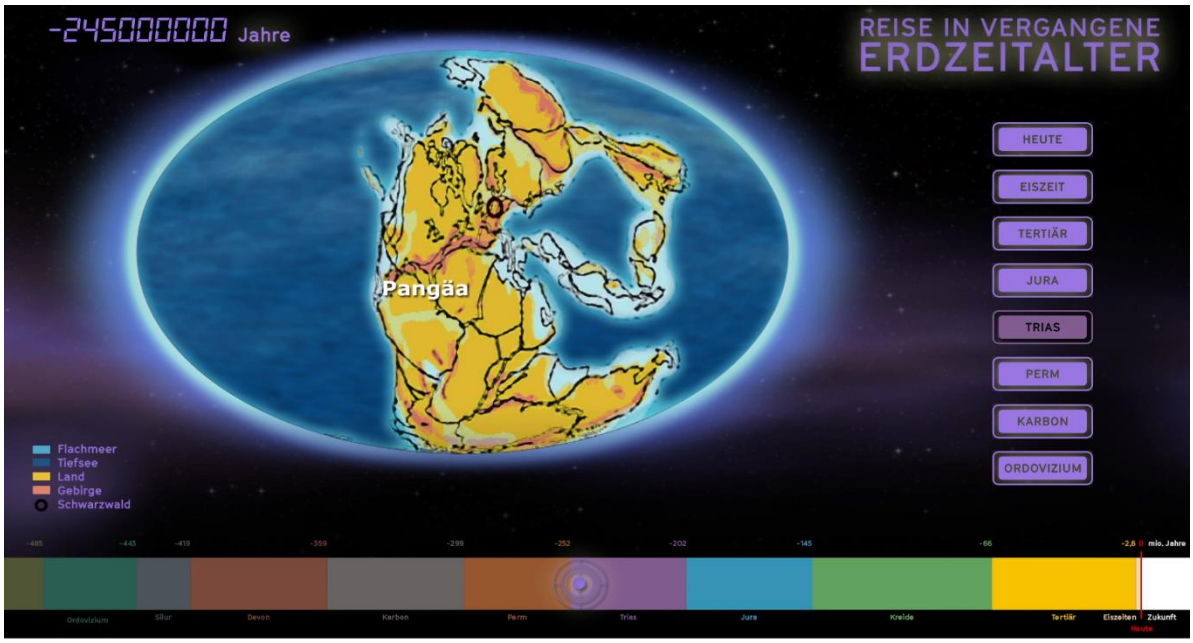


Abbildung 21: Benutzeroberfläche (Fofana 2015)

5. Steuerung der Geräte

Wie die einzelnen Geräte untereinander verbunden sind und wie sie angesteuert werden, wird in den nächsten Kapiteln erläutert.

5.1 Verknüpfung der Geräte

Die Verknüpfung der verschiedenen Geräte ist in einer technischen Skizze in Anhang C dargestellt. Anhang D zeigt die Positionierung der Geräte und die Kabelwege inner- und außerhalb der Kuppel.

Der Touch-PC ist über zwei Interfaces an die Geräte angebunden. Zum einen sorgt das USB DMX Interface für eine Signalübertragung zum Dimmer und Switcher um das Licht zu steuern, zum anderen hat das USB RS232 Interface die Aufgabe den Videoplayer mit Signalen zu steuern um die entsprechenden Video- und Audiodateien abzuspielen.

Die Geräte, welche sich auf der linken Seite der technischen Skizze befinden werden seitens der Veranstaltungstechniker und Elektriker verbaut. Dabei handelt es sich um passive Lautsprecher mit einem Verstärker, DMX- fähige Lampen und einen Bildschirm.

5.2 Steuerung der Geräte über Processing

Processing ist eine open source Software und eine Programmiersprache mit integrierter Entwicklungsumgebung, welche in der Programmierung von Sound, Animation und Bildern zum Einsatz kommt. Entwickelt wurde sie 2001 von Ben Fry und Casey Reas am *Massachusetts Institute of Technology* in Boston.

Die Software wird überwiegend von Künstlern genutzt und ermöglicht viele grafische Anwendungsmöglichkeiten.

Der Code in Processing basiert auf der Programmiersprache Java und kann auf zwei verschiedene Arten strukturiert werden: Funktionenorientiert oder objektorientiert. Es ist auch möglich beide Methoden zu kombinieren.
Funktionen beinhalten Code und werden immer dann aufgerufen, wenn sie benötigt werden. Beim Aufrufen der Funktion wird der Codeblock ausgeführt und kann dabei Variablen an andere Codesegmente übergeben.
Bei der Objektorientierung wird der Code über das Anlegen von Klassen in kleinere Teile, auch Objekte genannt, strukturiert (vgl. Greenberg 2007, S. 32–34).

Dabei versucht man den komplexen Code in ein Modell der realen Welt umzuwandeln. Die Objekte haben einen Zustand, ein Verhalten, eine Identität und können Methoden ausführen (vgl. Greenberg et al. 2013, S. 188–189).

Um Bilder, Schriftarten, Videos oder andere Dateien einbinden zu können, müssen diese in dem Ordner *data* abgelegt werden. Außerdem gibt es einen Ordner *library*, welcher alle heruntergeladenen Bibliotheken beinhaltet (vgl. ebd., S. 21).

Der weitere Code in Processing ist nach folgender Struktur aufgebaut:

Durch die *setup()-Funktion* wird das Programm initialisiert. Variablen können entweder vor oder in dieser Funktion deklariert werden um sie später im Code zu verwenden. Im Gegensatz zu der danach folgenden *draw()-Funktion*, wird diese nur einmal durchlaufen. Die *draw()-Funktion* besteht aus allen Inhalten, welche gezeichnet oder animiert werden und läuft in jedem Bild ab (vgl. Greenberg 2007, S. 78).

Aufgrund dieser Objektorientierung ist es vor allem für visuell denkende Menschen ein guter Einstieg in die Programmierung, was auch der Leitspruch von Processing aussagt:

> „Processing seeks to ruin the careers of talented designers by tempting them away from their usual tools and into the world of programming and computation. Similarly, the project is designed to turn engineers and computer scientists to less gainful employment as artists and designers.“ (Casey Reas & Ben Fry 2015b)

5.2.1 *setup()*

In der Funktion *setup()* werden alle Parameter einmalig festgelegt. Über *size* wird die Ausgabengröße bestimmt. Die grafische Oberfläche soll auf dem gesamten Bildschirm zu sehen sein.

Der Modus *Full Screen* wird über folgende Codezeilen erreicht.

```
void setup() {
  size(displayWidth, displayHeight, SMT.RENDERER);
[…]
}
boolean sketchFullScreen (){
 return true;
}
```

Die ersten beiden Variablen in *size* speichern den Wert der Bildschirmbreite und -höhe. Die Funktion *sketchFullScreen()* gibt den Wert *true* zurück und sorgt somit dafür, dass die Leiste am unteren Rand des Bildschirms nicht mehr sichtbar ist und das Programm im *Full Screen* Modus arbeitet.

5.2.2 Buttons

Die Oberfläche der Buttons wird über den Datentyp *PImage* realisiert. Zunächst werden einige Variablen für die später folgenden Funktionen angelegt.

```
//Buttonposition zu diesem Wert addieren (zum
//Verschieben)
float bPositionX = 0;
float bPositionY = 0;

//Bilder fuer Buttons
PImage btn_ordovizium_aktiv;
PImage btn_ordovizium_inaktiv;
PImage btn_karbon_aktiv;
PImage btn_karbon_inaktiv;
[…]
//aktuelles Button Bild
PImage currentImageOrdovizium;
PImage currentImageKarbon;
PImage currentImagePerm;
PImage currentImageTrias;
PImage currentImageJura;
PImage currentImagePalaeogen;
PImage currentImageHolozaen;
PImage currentImageHeute;
```

Die float- Variablen *bPositionX* und *bPositionY* bestimmen die Position der Buttons. Float ist ein elementarer Typ, welcher 32 Bit lange Fließkommazahlen akzeptiert (vgl. Sierra und Bates 2008, S. 51).

Durch diese Variablen ist es nicht nötig bei Anpassung der Oberfläche die festgelegte Position jedes einzelnen Buttons zu ändern, sondern nur die beiden Werte anzupassen. Anschließend werden für jedes Erdzeitalter zwei Variablen des Typs *PImage* erstellt. Die Variable *btn_ordovizium_aktiv* speichert das Bild, welches angezeigt werden soll, wenn der Button gedrückt wird und *btn_ordovizium_inaktiv* beinhaltet die Grafik für den Fall, dass keine Interaktion stattfindet. Abschließend erhält jedes Erdzeitalter eine Variable *currentImage[X]*. *X* ist in dieser Ausarbeitung ein Platzhalter für eines der acht Erdzeitalter. Der Variablen *currentImage[X]* wird jeweils einer der zwei zuvor genannten Variablen zugewiesen. Der Wert von *currentImage[X]* entspricht somit immer demjenigen Wert der gerade angezeigten Grafik.

Den 16 Variablen für die Anzeige des aktiven und inaktiven Buttons werden in der Funktion *setup()* über *loadImage()* die Grafiken zugeordnet.

```
btn_ordovizium_aktiv=
loadImage("btn_ordovizium_aktiv.png");
btn_ordovizium_inaktiv=
loadImage("btn_ordovizium_inaktiv.png");
```

Bei dem Start des Programms werden alle Buttons auf *inaktiv* gesetzt.

```
currentImageOrdovizium = btn_ordovizium_inaktiv;
currentImageKarbon = btn_karbon_inaktiv;
currentImagePerm = btn_perm_inaktiv;
currentImageTrias = btn_trias_inaktiv;
currentImageJura = btn_jura_inaktiv;
currentImagePalaeogen = btn_palaeogen_inaktiv;
currentImageHolozaen = btn_holozaen_inaktiv;
currentImageHeute = btn_heute_inaktiv;
```

5.2.3 Bibliotheken

Eine Bibliothek dient der Software als Ergänzung. Sie ermöglicht in spezifischen Bereichen wie beispielsweise Video oder Audio die Nutzung von weiteren Funktionen.

5.2.3.1 *Simple Multi-Touch*

Die Bibliothek *Simple Multi-Touch*, auch SMT genannt, wurde entwickelt um Gesten mit einem oder mehreren Fingern auf einer berührbaren Bildschirmoberfläche zu ermöglichen.
SMT wird in Processing über den Befehl *SMT.init()* gestartet.
Anschließend werden Zonen erstellt. Eine Zone bildet den Bereich, auf welchem man ein Objekt auf dem Bildschirm berühren kann. In der *draw()-Methode* dieser Zone werden die Objekte gemalt.
Die *touch()-Methode* legt die Reaktion auf einen Kontakt mit dieser Zone fest.
Die *pressed()-Methode* verarbeitet ebenfalls Berührungen. Sie verhält sich allerdings wie ein Knopf, welcher gedrückt wird und behält den Zustand somit auch bei, wenn es keine Berührung mehr an dieser Stelle gibt (vgl. Simple Multi-Touch (SMT) Toolkit for Processing[Home]).

Sobald SMT initialisiert ist, werden in der *setup()-Funktion* von Processing die acht Zonen für die Erdzeitalter erstellt.

```
//Zonen fuer die Touch-Events erstellen
  Zone ordoviziumzone = new Zone( "OrdoviziumZone");
  ordoviziumzone.translate(1600, 815);
  SMT.add( ordoviziumzone);
  Zone karbonzone = new Zone( "KarbonZone");
  karbonzone.translate(1600, 720);
  SMT.add( karbonzone);
  Zone permzone = new Zone( "PermZone");
  permzone.translate(1600, 625);
  SMT.add( permzone);
  Zone triaszone = new Zone( "TriasZone");
  triaszone.translate(1600, 530);
  SMT.add( triaszone);
  Zone jurazone = new Zone( "JuraZone");
  jurazone.translate(1600, 435);
  SMT.add( jurazone);
  Zone palaeogenzone = new Zone( "PalaeogenZone");
  palaeogenzone.translate(1600, 340);
  SMT.add( palaeogenzone);
  Zone holozaenzone = new Zone( "HolozaenZone");
  holozaenzone.translate(1600, 245);
  SMT.add( holozaenzone);
  Zone heutezone = new Zone( "HeuteZone");
  heutezone.translate(1600, 150);
  SMT.add( heutezone);
```

In den Apostrophen steht der Name der jeweiligen Zone, damit diese im späteren Verlauf angesprochen und bearbeitet werden kann.

Über den Befehl *translate* wird festgelegt, an welcher Stelle das Koordinatensystem der jeweiligen Schaltfläche beginnt. Der erste Wert gibt die x-Koordinate und der zweite Wert die y-Koordinate an. Wie man in folgender Abbildung sehen kann, wird das Koordinatensystem der Zone *Holozän* bei *x*= 1600 und *y*= 245 angelegt und hat somit bei diesem Wert ihren Ursprungspunkt (Abbildung 22).

Abbildung 22: Koordinatensystem der Zone Holozän (Eigene Darstellung)

Im nächsten Schritt werden außerhalb der *setup()*- und *draw()- Funktion* weitere Funktionen angelegt, welche die Schaltflächen für die jeweilige Zone erstellen und den Namen *draw[Zonenname]* erhalten.

```
void drawHolozaenZone( Zone zone) {
  //Button Holozaen
Image (currentImageHolozaen,0+bPositionX,0+bPositionY);
}
```

In dieser Funktion kann man die Oberfläche des Buttons gestalten. Diese kann beispielsweise ein Rechteck sein oder, wie in dieser Installation, als Bild angelegt werden. Die beiden nummerischen Werte geben an, auf welcher x- und y-Koordinate die Oberfläche des Buttons in der Zone ihren Ursprung haben soll. Da die Variablen *bPositionX* und *bPositionY* am Anfang des Codes auf 0 gesetzt werden, liegt das Bild für die Schaltfläche *Holozän* bei der x-Koordinate 0 und y-Koordinate 0, auf das gesamte Koordinatensystem von Processing betrachtet bei 1600 und 245, da die Zone zuvor an dieser Stelle angelegt wurde (Abbildung 23).

Abbildung 23: Koordinatensystem der Zone Holozän (Button Eiszeit) mit Urspung (0/0) (Eigene Darstellung)

Die nächste Methode verarbeitet Berühr-Ereignisse und wird als *touch[Zonenname]* bezeichnet.

```
void touchHolozaenZone( Zone zone) {

}
```

Diese Aktion wird nur für den Moment der Berührung ausgeführt und danach sofort wieder verworfen. Bei Berührung der Schaltfläche *Holozän* soll keine Aktion ausgeführt werden. Dasselbe gilt auch für die restlichen Buttons.

Eine weitere Funktion ist *press[Zonenname]*, welche bei Berührung einen Zustand annimmt und diesen im Gegensatz zu den *Touch()- Funktionen* beibehält.

Hier werden drei Komponenten gesteuert: Die Animation, der Videoplayer mit den Video- und Tondateien und der Dimmer beziehungsweise Switch für die Lichtsteuerung. Der Aufbau des Codes zur Steuerung dieser Komponenten wird in den folgenden Kapiteln detaillierter erklärt.

Des Weiteren wird in dieser Funktion auch die Buttonoberfläche angepasst. Die Variable *currentImage[X]* wird für den betätigten Button auf *aktiv* gesetzt, während die restlichen den Wert *inaktiv* zugewiesen bekommen.

```
currentImageOrdovizium = btn_ordovizium_aktiv;
currentImageKarbon = btn_karbon_inaktiv;
currentImagePerm = btn_perm_inaktiv;
currentImageTrias = btn_trias_inaktiv;
currentImageJura = btn_jura_inaktiv;
currentImagePalaeogen = btn_palaeogen_inaktiv;
currentImageHolozaen = btn_holozaen_inaktiv;
currentImageHeute = btn_heute_inaktiv;
```

5.2.3.2 *video*

Durch die Verwendung der Bibliothek *video* kann man in Processing Videos in den Code einbinden und anzeigen. Sie besteht aus zwei Klassen: *Movie* und *Capture*. Die Klasse *movie* lädt die Videos an bestimmten Stellen, kann sie stoppen, wiederholen oder die Geschwindigkeit verändern. Die Klasse *capture* kann Videos von einem angeschlossenen externen Gerät anzeigen (vgl. Casey Reas & Ben Fry 2015c). Da diese Funktion für die Installation nicht notwendig ist, wird sie nicht eingesetzt.

Zu Beginn werden Variablen des Typs *boolean* angelegt. Diese können entweder den Wert *true* oder *false* annehmen.

```
boolean ordoviziumPressed=false;
boolean karbonPressed = false;
boolean permPressed = false;
boolean triasPressed= false;
boolean juraPressed = false;
boolean palaeogenPressed = false;
boolean holozaenPressed = false;
boolean heutePressed = false;
```

Die obenstehenden Variablen *[Name]Pressed* geben den aktuell berührten Button an, während die unten stehenden *[Name]Pressedbefore* denjenigen angeben, welcher als letztes vor dem aktuellen berührt wurde.

```
boolean   ordoviziumPressedbefore = false;
boolean   karbonPressedbefore = false;
boolean   permPressedbefore = false;
boolean   triasPressedbefore = false;
boolean   juraPressedbefore = false;
boolean   palaeogenPressedbefore = false;
boolean   holozaenPressedbefore = false;
boolean   heutePressedbefore = false;
```

Zudem werden noch Variablen für die Standbilder angelegt. Diese zeigen das jeweilige Erdzeitalter mit Beschriftung der Kontinente.

```
//Standbilder
PImage standbildOrdovizium;
PImage standbildKarbon;
[…]

boolean showStandbildOrdovizium = false;
boolean showStandbildKarbon = false;
[…]
```

Die Variablen des Typs *boolean* sind zu Beginn auf *false* gesetzt und beschreiben, ob das Standbild gerade angezeigt werden soll oder nicht.

Des Weiteren wird ein *Movie* angelegt, welchem die Animation der Kontinentaldrift zugewiesen wird.

In der *setup()- Funktion* wird die Bildfrequenz festgesetzt. Sie gibt an wie viele Bilder pro Sekunde gelesen werden (vgl. Casey Reas & Ben Fry 2015c). Die Bildsequenz erhält den Wert 30 um Verzögerungen beim Abspielen der Animation zu verhindern.

```
frameRate(30);
```

Anschließend wird die Animation in das Programm geladen.

```
drift = new Movie(this, "drift.mov");
```

In der *draw()- Funktion* wird diese abgespielt und gesteuert.

Über die folgenden Codezeilen wird getestet ob das nächste Frame zum Lesen bereit ist. Anschließend wird der Alpha Kanal auf 100% angepasst und die Ausgabeposition bestimmt.

```
//Animation lesen
  if (drift.available()) {
    drift.read();
  }
//Alpha Kanal 100%
  tint(255, 255);
//Animation positionieren
  image(drift, 0, 0);
```

Die Kontinentaldrift soll jeweils vom letzten berührten Button zur aktuell gedrückten Schaltfläche ablaufen. Wird das Erdzeitalter *Perm* ausgewählt und anschließend *Ordovizium*, soll man die Verschiebung der Platten vom Erdzeitalter *Perm* nach *Ordovizium* sehen.

Um dies im Programmiercode umzusetzen nutzt man die bereits oben erwähnten *boolean* Variablen *[Zonenname]Pressedbefore* und *[Zonenname]Pressed* und *if-Abfragen*. Diese prüfen über ein Schlüsselwort ob eine Bedingung erfüllt ist oder nicht und führen dementsprechend einen Code aus (vgl. Greenberg et al. 2013, S. 74).

Für die Erdzeitalter gibt es acht *if- Abfragen*, welche wiederum sieben geschachtelte *if- Abfragen* beinhalten. Die übergeordneten *if- Abfragen* prüfen, welche Schaltfläche soeben berührt wurde. In der Funktion *press[Zonenname]* wird die Variable *[Zonenname]Pressed* bei Aufruf auf den Wert *true* gesetzt und gibt somit die Information an die *if- Abfrage* weiter. Den restlichen Variablen wird der Wert *false* zugewiesen damit die jeweilige if-Abfrage in der Methode *draw()* nicht in einer Endlosschleife festhängt.

```
void pressHolozaenZone( Zone zone) {
     […]
ordoviziumPressed=false;
karbonPressed = false;
permPressed = false;
triasPressed= false;
juraPressed = false;
palaeogenPressed = false;
heutePressed = false;
holozaenPressed=true;

}
```

Sobald abgefragt wird welcher Button aktuell betätigt wird, befindet sich der Code in den geschachtelten *if- Abfragen*. Diese prüfen, welche Schaltfläche zuvor berührt wurde um die Animation an der richtigen Stelle zu beginnen. Hierfür wird wieder die Funktion *press[Zonenname]* verwendet.

Bei Druck auf eine Schaltfläche wird im ersten Codeabschnitt die Variable *[Zonenname]Pressed* abgefragt. Wenn diese auf *true* gesetzt ist, wird auch die jeweilige Variable *[Zonenname]Pressedbefore* auf *true* gesetzt und die Animation springt auf den Zeitpunkt des Erdzeitalters.

Die Tabelle zeigt ab welchem Zeitpunkt das Erdzeitalter angesteuert und abgespielt werden muss. Die Animation läuft vom ältesten Erdzeitalter bis hin zu heute und im Anschluss nochmals gespiegelt von heute bis zum ältesten Erdzeitalter ab.

Erdzeitalter	**Vorwärts in Sekunden**	**Rückwärts in Sekunden**
Ordovizium	3	86
Karbon	17	71
Perm	20	68
Trias	23	65
Jura	30	59
Paläogen/Tertiär	42	47
Holozän/Eiszeit	45	45
Heute	45	45

Tabelle 3: Ansteuerung der Kontinentaldrift-Animation (Eigene Darstellung)

Zur Veranschaulichung folgt nun der hierfür relevante Codeabschnitt der Funktion *press[Zonenname]*.

```
void pressHolozaenZone( Zone zone) {

//Schleifen testen welcher Button zuvor gedrueckt wurde
//damit die Animation von dort aus starten kann

  if (ordoviziumPressed==true) {
    ordoviziumPressedbefore = true;
    //Animation an der jeweiligen Stelle abspielen
    drift.jump(3);
    drift.play();
  }

  if (permPressed==true) {
    permPressedbefore=true;
    //Animation an der jeweiligen Stelle abspielen
    drift.jump(20);
    drift.play();
  }
[…]
holozaenPressed=true;
}
```

Hier wird die Schaltfläche *Holozän* berührt. Anschließend wird über *if- Abfragen* geprüft, welcher Button zuvor gedrückt wurde. Wenn es *Ordovizium* war, ist *ordoviziumPressed* auf *true* gesetzt, kommt dadurch in die *if- Abfrage* und setzt die Variable *ordoviziumPressedbefore* auf *true*. Am Ende des Durchlaufs der Funktion wird die Variable *holozaenPressed* auf *true* gesetzt um bei der nächsten Berührung einer Schaltfläche auf dieselbe Weise verarbeitet werden zu können.

Die Animation muss nicht nur in ihrem Startzeitpunkt, sondern auch in der Abspiellänge gesteuert werden. Dies geschieht in der *draw()- Funktion*. Über *drift.time()* wird der aktuelle Zeitpunkt der Animation als *float*- Wert zurückgegeben. Der angelegte *integer*- Wert *timestamp* multipliziert diesen mit 10000 um in der weiteren Programmierung mit *integer*- anstatt *float*- Werten arbeiten zu können.

```
timestamp = int(drift.time()*10000);
```

Über folgende *if- Abfrage* wird getestet bei welchem Zeitstempel sich die Animation befindet. Sobald der jeweilige Zeitpunkt erreicht ist, wird die Animation pausiert. Hier hält die Animation zwischen dem Zeitpunkt 86 und 87 Sekunden an. Daraufhin wird der *boolean*- Wert zur Anzeige des Standbildes auf *true* gesetzt und *[X]pressedBefore* auf *false*, damit die *if- Abfrage* verlassen wird und keine Endlosschleife entsteht.

```
if (timestamp>869999 && timestamp<879999) {
        drift.pause();
        showStandbildOrdovizium = true;
        karbonPressedbefore = false;
 }
```

Sobald die *if- Abfrage* über die zuvor betätigten Buttons betreten wird, werden alle Variablen *showStandbild[X]* auf *false* gesetzt und erst wieder bei einer Pause der Animation einem anderen Wert zugewiesen.

Erhält eine Variable *showStandbild[X]* den Wert *true*, wird eine weitere *if- Abfrage* ausgelöst. Diese bewirkt das Anzeigen des jeweiligen Standbildes.

```
if (showStandbildOrdovizium == true) {
      image(standbildOrdovizium, 0, 0);
    }
```

Unter der Kontinentaldrift befindet sich ein ebenfalls animierter Zeitstrahl, welcher eine Übersicht über die acht Erdzeitalter und den aktuell ausgewählten Zeitpunkt bietet (Abbildung 25). Der Zeitstrahl läuft mit der Animation mit und wird somit über die zuvor erläuterten Codezeilen mitgesteuert (Abbildung 24).

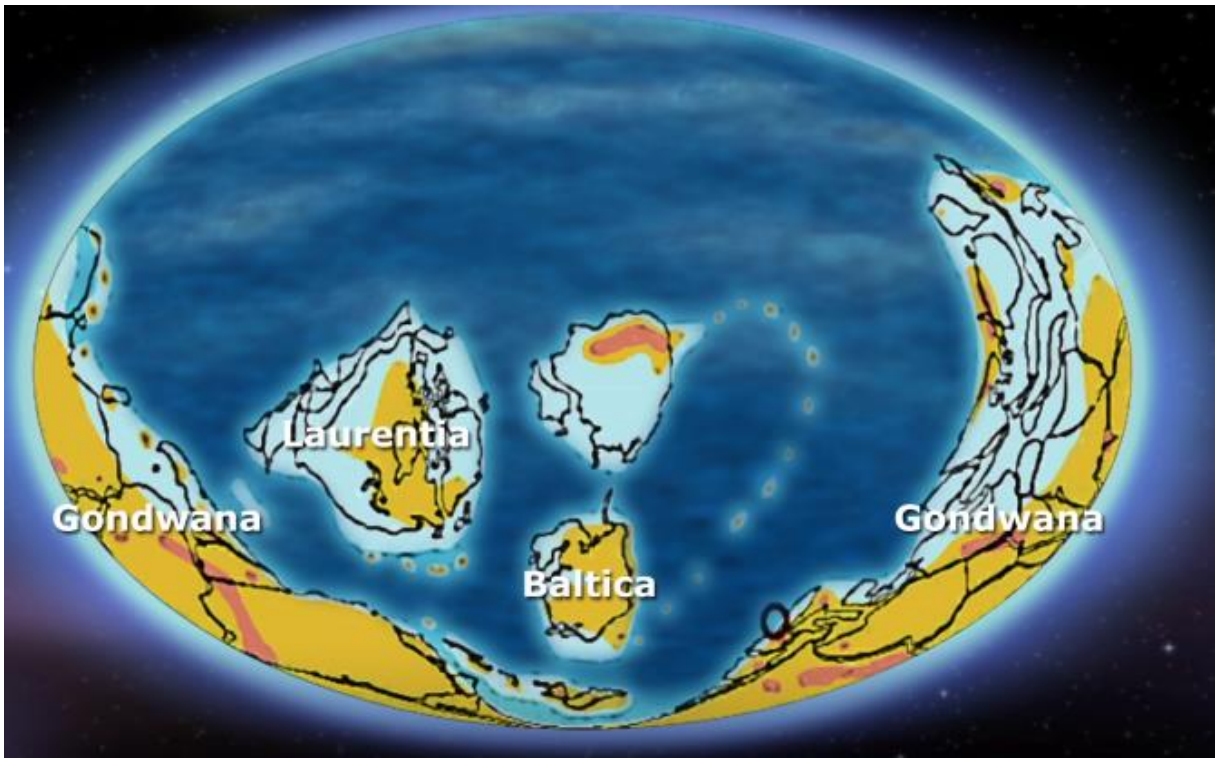


Abbildung 24: Animation der Kontinentaldrift (Fofana 2015)

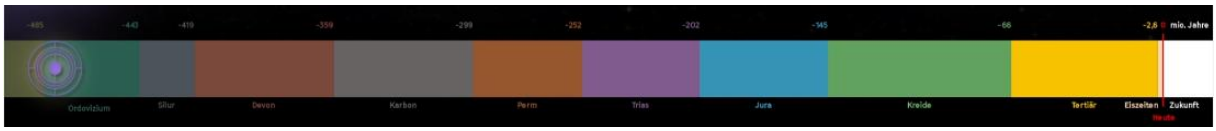


Abbildung 25: Zeitstrahl (Fofana 2015)

5.2.3.3 *dmxP512*

Die *dmxP512- Bibliothek* ermöglicht es über eine serielle Schnittstelle Signale und Befehle von Processing an das DMX USB Interface zu senden. Dieses ist zuständig für die Ansteuerung der Lampen, welche die Gesteine und Fossilien beleuchten.

Mit der String Variablen *DMXPRO_PORT* wird der COM Port, über welchen das Interface angesprochen wird, festgelegt. Der integer-Wert *DMXPRO_BAUDRATE* legt die Baudrate und die *universeSize* die Anzahl an Kanälen fest.

Über die Codezeile *dmxOutput.set(channel,value)* wird einem Kanal ein bestimmter Wert zugewiesen und so an das Interface übergeben (vgl. dmxP512 : DMX processing.org library 2015).

Der *DMXPRO_PORT* wird dem Port *COM3* zugewiesen, da dies bei der Konfiguration des *DMX-USB Pro Mk2* festgelegt wurde.

Über Processing ist es auch möglich sich in der Konsole eine Liste aller verfügbaren COM Ports anzeigen zu lassen.

```
//alle verfuegbaren COM Ports in der Konsole auflisten
   println(Serial.list());
```

Die Anzahl an Kanälen beträgt acht, da es nur acht verfügbare Channels gibt.

Die Baudrate wird wie in der Gerätedokumentation empfohlen auf 115000 Bit/s festgesetzt.

Die folgende Codezeile weist dem *Output* in der *setup()- Funktion* die zuvor festgelegten Werte zu.

```
dmxOutput.setupDmxPro(DMXPRO_PORT, DMXPRO_BAUDRATE);
```

In der Funktion *press[Zonenname]* werden den Kanälen die entsprechenden Werte zugewiesen. Bei jedem Erdzeitalter soll ein anderes Gestein beleuchtet werden, folglich eine andere Lampe aufleuchten.

Wie schon in den vorherigen Kapiteln erläutert, erhält jedes DMX- fähige Gerät eine Startadresse.

Für diese Installation sieht die Adressierung folgendermaßen aus:

Gerät	**Kanäle gesamt**	**Channel**	**Belegter Adressbereich**	**Steuert an**	**Auslösendes Erdzeitalter**
Switch	4	1	1	Lampe 1	Ordovizium
		2	2	Lampe 2	Karbon
		3	3	Lampe 3	Perm
		4	4	Lampe 4	Trias
Dimmer	4	1	5	Lampe 5	Jura
		2	6	Lampe 6	Tertiär
		3	7	Lampe 7	Eiszeit
		4	8	-	

Tabelle 4: Adressierung der Geräte in der Zeitmaschine (Eigene Darstellung)

Berührt der Benutzer den Button *Eiszeit*, soll Lampe 7 aufleuchten. Channel 1 bis 6 erhalten den Wert 0, da Licht 1 bis 6 nicht leuchten soll und Channel 7 wird auf den Wert 255 gesetzt, welcher die Lampe auf die stärkste Leuchtkraft setzt.

```
//das erste ist der Channel und das zweite der Wert
dmxOutput.set(1, 0);
dmxOutput.set(2, 0);
dmxOutput.set(3, 0);
dmxOutput.set(4, 0);
dmxOutput.set(5, 0);
dmxOutput.set(6, 0);
dmxOutput.set(7, 255);
```

5.2.3.4 *serial*

Das Einbinden der Bibliothek *serial* ermöglicht es Daten über den seriellen Port zu senden und zu empfangen.

Für die Programmierung der Zeitmaschine wird *serial* als Kommunikationsmittel zwischen Processing und dem Videoplayer genutzt. Dieser wiederum spielt je nach Signal ein Landschaftsvideo und eine Audiodatei ab.

Im *setup()* wird der angelegte serielle Port initialisiert.

```
void setup(){
    […]
    mySerialPort = new Serial(this, Serial.list()[2],
    serial_baudrate);
}
```

Wie bereits bei der Initialisierung des DMX Interface werden die Portbelegung und die Baudrate aus der Gerätedokumentation benötigt. Diese liegt bei 4800 Bit/s und die Belegung auf COM7 beziehungsweise an der dritten Stelle in der Auflistung der Ports.

Um die Videos und den Ton steuern zu können, werden für jedes Erdzeitalter jeweils zwei Variablen *serial[X] und stopserial[X]* des Typs *boolean* angelegt und als Anfangswert auf *false* gesetzt.

```
//Landschafts-Video triggern
boolean serialOrdovizium = false;
[…]
//Video und Audio pausieren
boolean stopserialOrdovizium = false;
[…]
```

Sobald eine Schaltfläche gedrückt wird, wird die Funktion *press[Zonenname]Zone* aufgerufen und setzt die jeweilige Variable serial[X] auf *true*, während die restlichen den Wert *false* beibehalten.

```
void pressOrdoviziumZone( Zone zone) {
[…]
  //Landschafts-Video triggern
  serialOrdovizium = true;
  serialKarbon = false;
  serialPerm = false;
  serialTrias = false;
  serialJura = false;
  serialPalaeogen = false;
  serialHolozaen = false;
  serialHeute = false;
}
```

Der Befehl *millis()* gibt die verstrichene Zeit seit Start des Programms in Millisekunden zurück (vgl. Casey Reas & Ben Fry 2015a). Hier speichert die *integer* Variable *serialstart* diese Zeit bei Eintritt in die Funktion.

```
serialstart = millis();
```

Diese Zeitangabe wird in der Methode *draw()* weiterverarbeitet. Sobald das Standbild angezeigt wird, soll das Landschaftsvideo mit einem Ankunftston und Sprechertext im Hintergrund ablaufen. Dies geschieht indem über eine *if- Abfrage* alle Variablen *serial[X]* überprüft werden und falls der Wert durch Druck auf den Button auf *true* gesetzt ist, wird die *if- Abfrage* ausgeführt.

Der Befehl *mySerialPort.write()* schickt den Wert in der Klammer an den Videoplayer.

Der Wert 1 spielt die erste Datei auf der im Videoplayer befindlichen SD- Karte ab und der Wert 2 spielt die zweite Datei ab. Dieses Verfahren funktioniert für das Senden und Ansprechen der restlichen Daten auf dieselbe Art. Die folgende Tabelle zeigt welche Datei über den seriellen Port angesteuert wird.

mySerialPort(X)	**angesteuerte Datei**	**Dateiart**
X=1	1_ordovizium	Video mit Audio
X=2	2_karbon	Video mit Audio
X=3	3_perm	Video mit Audio
X=4	4_trias	Video mit Audio
X=5	5_Jura	Video mit Audio
X=6	6_tertia	Video mit Audio
X=7	7_eiszeit	Video mit Audio
X=8	8_heute	Video mit Audio
X=9	9_losfliegenUndFliegen	Audio

Tabelle 5: angesteuerte Dateien über RS232 (Eigene Darstellung)

In der *if- Abfrage* wird der Wert *stopserial[X]* auf *true* gesetzt. Die Rechnung *millis()-serialstart* überprüft wie lange das Video seit dem Start schon abgespielt wird. Sobald die Zeit 150000 Millisekunden erreicht ist, wird eine weitere *if- Abfrage* ausgeführt. Diese pausiert das Video, setzt *serialStart* wieder auf die aktuelle Zeit und *stopserialOrdovizium* auf *false* um die *if- Abfrage* wieder zu verlassen.

```
//Landschaftsvideo mit Audiospur ansteuern
  if (serialOrdovizium == true) {
     mySerialPort.write(1);
    serialOrdovizium = false;
    stopserialOrdovizium = true;
  }
//Sind schon 150000 ms nach Druck auf den Button
//verstrichen? Wenn ja, Video und Audio pausieren
  if (stopserialOrdovizium == true && (millis()-
serialstart)>150000) {
//235 hat den Hex Wert EB (Datei pausieren)
    mySerialPort.write(235);
    serialstart = millis();
    stopserialOrdovizium = false;
  }
```

Die Datei wird über den HEX Wert *EB* pausiert und zeigt ein Standbild an. Dieser Wert entspricht dem Dezimalwert 235 und wird ebenfalls über die Codezeile *mySerialPort.write()* an den Videoplayer gesendet (vgl. mathe-lexikon.at).

Bevor das Landschaftsvideo abgespielt wird, soll die Animation der Kontinentaldrift mit einem Start- und Reisegeräusch ablaufen. Mit dem Auslösen der Animation wird über den seriellen Port ein Signal an den Videoplayer gesendet, welches die Audiodatei triggert.

Diese Tonspur wird bei allen Reisen bis auf diejenige zwischen *Eiszeit* und *Heute* ausgelöst, da diese durch ihre kurze Reisezeit von einer Sekunde weder ein Start- noch ein Reisegeräusch benötigen.

```
//Touch Pressed Funktion fuer die "HeuteZone"
void pressHeuteZone( Zone zone) {
[…]
  if (ordoviziumPressed==true) {
    ordoviziumPressedbefore = true;
//Animation an der jeweiligen Stelle abspielen und
Audiospur (Losfliegen und fliegen) abspielen
    mySerialPort.write(9);
    drift.jump(3);
    drift.play();
  }
}
```

Die Audiodatei wird mit der Animation an dem entsprechenden Zeitpunkt pausiert.

```
//Bei diesem Zeitpunkt soll die Animation anhalten
      if (timestamp>299999 && timestamp<309999) {
        //Audio Losfliegen und Fliegen pausieren
        mySerialPort.write(235);
        //Animation anhalten
        drift.pause();
        //Standbild soll angezeigt werden
        showStandbildJura = true;
        karbonPressedbefore = false;
      }
```

6. aufgetretene Probleme

Bei der Planung und Umsetzung der Installation gab es einige Probleme, welche im Folgenden näher erläutert werden.

6.1 Ansteuerung des Videoplayers

Die Ansteuerung des Videoplayers erfolgt über einen Datenaustausch über die serielle Schnittstelle. Processing sendet bei Druck auf einen Button den entsprechenden Befehlswert als Zeichenfolge an den Videoplayer, welcher dadurch die jeweilige Datei abspielen soll. Jedoch kann der Videoplayer diese ankommenden Daten nicht verarbeiten und reagiert somit nicht. Den Ursprung dieses Problems kann man über den Einsatz einer weiteren Software ausfindig machen. Durch das Anlegen eines virtuellen seriellen Ports kann man sehen welche Daten Processing abschickt. Hierbei fällt auf, dass der Befehl an den Videoplayer bei Druck auf eine Schaltfläche nicht nur einmal, sondern mehrmals hintereinander abgeschickt wird. Aus diesem Grund ist weiterer Code notwendig, welcher das einmalige Senden der Zeichenfolge garantiert.

6.2 Dimmen der Lampen

Die Lampen in der Zeitmaschine sollen über einen Dimmer angesteuert werden und wie im Folgenden beschrieben das Licht langsam auf 100% setzen.

Berührt der Benutzer den Button *Eiszeit*, soll Lampe 7 aufleuchten. Channel 1 bis 6 erhalten den Wert 0, da Licht 1 bis 6 nicht leuchten soll und Channel 7 wird über eine *for- Schleife* auf den Wert 255 gesetzt. Die *for- Schleife* hat den Effekt, dass das Licht nicht umgehend auf die volle Leuchtstärke gesetzt wird. In 16 Durchläufen wird es jeweils um den Wert 15 erhöht bis es den Maximalwert 255 erreicht.

```
for (int i=0; i<16; i=i+15) {
    //das erste ist der channel und das zweite der wert
    dmxOutput.set(1, 0);
    dmxOutput.set(2, 0);
    dmxOutput.set(3, 0);
    dmxOutput.set(4, 0);
    dmxOutput.set(5, 0);
    dmxOutput.set(6, 0);
    dmxOutput.set(7, i);
}
```

Durch die Auswahl von LED Lampen als Lichtquelle ist es jedoch nicht mehr möglich diese zu Dimmen. LEDs arbeiten mit Gleichstrom und nicht mit Wechselstrom, welcher für das Dimmen von Licht eine Voraussetzung ist. Folglich wird der Wert nicht über mehrere Schritte, sondern unvermittelt auf 100% gesetzt.

6.3 Bibliothek *video*

Die Animation wird über die Bibliothek *video* eingebunden und soll nach Dokumentation auch die Möglichkeit des Rückwärtsabspielens bieten. Diese Funktion wird benötigt, da der Benutzer nicht nur in die Vergangenheit reisen kann, sondern auch von der Vergangenheit in eine näher an der Gegenwart gelegene Zeit. Das Rückwärtsabspielen funktioniert jedoch in keinem der angegeben Videoformate. Dieses Problem wird gelöst, indem die Animation in gespiegelter Form an das Ende der ursprünglichen Datei gesetzt wird und somit bei Ansteuerung alles vorwärts abgespielt werden kann.

6.4 Bibliothek *Simple Multi-Touch*

Die Funktion *mousePressed()* soll eine Interaktion zwischen Processing und dem Benutzer ermöglichen. Der Einsatz dieser ist nicht nur für Mausklicks, sondern auch bei Druck einer Berührоberfläche möglich. Die aufkommende Problematik hierbei ist, dass nur ein einfacher Druck nicht reicht. Damit der Programmcode die Funktion *mousePressed()* ausführt ist ein Doppelklick notwendig. Dieses bedienungsunfreundliche Userinterface kann man über die Bibliothek *Simple Multi-Touch* umgehen, da sie vor allem für berührungsempfindliche Benutzeroberflächen ausgelegt ist.

6.5 Multi-Threading

Mit dem Einsatz der Bibliothek *Simple Multi-Touch* entsteht ein weiteres Problem. Der standardmäßig verwendete Render Modus *P2D* oder *P3D* muss auf *SMT.RENDERER* umgestellt werden.

Threads wiederum können nur in dem Modus *P2D* oder *P3D* eingebunden werden. Durch Multi-Threading wird das gleichzeitige Abarbeiten mehrerer Prozesse ermöglicht und verhindert somit Performanceprobleme. Da es nicht möglich ist, Threads zu verwenden, werden alle Prozesse in einem main- Thread abgespielt. Das Auslösen mehrerer Aktionen sollte bei einem Programmcode diesen Umfangs jedoch keine Performanceprobleme bereiten.

7. Fazit

Trotz der aufgetretenen Probleme ist es gelungen aus den einzelnen Elementen ein multimediales Gesamtwerk zu konzipieren und umzusetzen.

Hierbei war es eine große Herausforderung Geräte ausfindig zu machen, welche zum einen über die Software Processing ansteuerbar sind und zum anderen die gewünschten Aktionen ausführen können. Außerdem kommt noch hinzu, dass sie sich untereinander nicht in der Kommunikation stören dürfen. Sobald diese drei Forderungen erfüllt sind, liegt ein weiterer Schwerpunkt auf der Planung des Ablaufs. Dieser soll sicherstellen, dass alle durch den Benutzer ausgelösten möglichen Szenarien bedacht werden, wie beispielsweise was bei der Unterbrechung einer Reise passiert, ab wann die Tonspur einsetzen soll oder wie lange die Landschaftsanimation angezeigt werden soll.

Zusammenfassend ist es wichtig neben der technischen Planung den Überblick über alle durch den Benutzer ausgelösten Eventualitäten zu behalten und diese in der Umsetzung zu bedenken.

Oftmals gerät die Technik bei solchen Installationen in den Hintergrund, da der Besucher nur mit der Benutzeroberfläche in Berührung kommt. Trotzdem sind die technische Planung und praktische Umsetzung ein essentieller Teil, geradezu der Grundbaustein einer Installation.

Ein großer Vorteil dieser Installation ist die breite Zielgruppe. Auch wenn die Zeitkapsel altersgerecht für die Kinderabteilung gestaltet ist, bietet sie ein Erlebnis für Jung und Alt. Vor allem durch das Aktivieren der verschiedenen Sinne werden unterschiedliche Lerntypen angesprochen. Diese innovative und für den normalen Museumsalltag ungewöhnliche Art Wissen zu vermitteln ermöglicht den Besuchern sich noch lange Zeit an das Erlernte zu erinnern.

Die berührbare Oberfläche ist vor allem für Kinder sehr spannend und erfordert eine intensivere und zugleich spielerischere Auseinandersetzung mit dem Thema als eine Texttafel zum Lesen. Durch die Interaktion mit dem Medium werden die Besucher des Museums aktiv zum Ausprobieren und Nachdenken angeregt.

Abschließend lässt sich sagen, dass vor allem in der heutigen Gesellschaft eine Installation dieser Art und somit der Einsatz von Multimedia und Interaktion von hoher Bedeutung ist, um Langeweile zu verhindern und die Besucher immer wieder aufs Neue zu faszinieren und zu begeistern.

Abbildung 26: Besucher der Zeitmaschine (Schlessmann 2015)

Literaturverzeichnis

13db Audio Education. Online verfügbar unter https://13db.de/, zuletzt geprüft am 23.06.2015.

Amonstar. Online verfügbar unter http://www.amonstar.com/eshop/, zuletzt geprüft am 11.05.2015.

Bewer, Rainer; Steckmann, Kai (2004): Das Praxisbuch der Lichttechnik. Einführung in die professionelle Bühnenbeleuchtung. 2., überarb. und erw. Aufl. München: Carstensen (Factfinder-Serie).

Box, Harry C. (2010): Set lighting technician's handbook. Film lighting equipment, practice, and electrical distribution. 4th ed. Burlington, Mass., London (U.K.): Focal Press.

Burghardt, Frank (Hg.) (2009): Lichttechnik für Einsteiger. Die eigene Lightshow mit DMX professionell steuern. 1. Aufl. Aachen: Elektor-Verl.

Burmester, Michael; Görner, Claus; Maly, Julia: Usability für Kids. So testet und gestaltet man interaktive Medien für Kinder.

Cameo Light. Online verfügbar unter http://www.cameolight.com/, zuletzt geprüft am 31.03.2015.

Casey Reas & Ben Fry (2015a): millis \ Language (API) \ Processing 2+. Processing. Online verfügbar unter https://processing.org/reference/millis_.html, zuletzt aktualisiert am 29.01.2015, zuletzt geprüft am 08.05.2015.

Casey Reas & Ben Fry (2015b): Overview \ Processing.org. Processing. Online verfügbar unter https://processing.org/overview/, zuletzt aktualisiert am 29.01.2015, zuletzt geprüft am 02.04.2015.

Casey Reas & Ben Fry (2015c): Video \ Libraries \ Processing.org. Processing. Online verfügbar unter https://www.processing.org/reference/libraries/video/, zuletzt aktualisiert am 28.04.2015, zuletzt geprüft am 07.05.2015.

Conrad. Online verfügbar unter http://www.conrad.de/ce/, zuletzt geprüft am 11.05.2015.

Daisy-Chaining :: daisy chaining :: ITWissen.info (2004). Online verfügbar unter http://www.itwissen.info/definition/lexikon/Daisy-Chaining-daisy-chaining.html, zuletzt aktualisiert am 31.03.2015, zuletzt geprüft am 31.03.2015.

designandtechtheatre on WordPress.com. Online verfügbar unter https://designandtechtheatre.wordpress.com/, zuletzt geprüft am 11.05.2015.

dmxP512 : DMX processing.org library (2015). Online verfügbar unter http://motscousus.com/stuff/2011-01_dmxP512/, zuletzt aktualisiert am 26.02.2015, zuletzt geprüft am 18.04.2015.

Ebner, Michael (2011): Lichttechnik für Bühne und Disco. Ein Handbuch für Praktiker. 7. Aufl., neu und überarb. Aachen: Elektor-Verl.

Enttec - DMX USB Pro Interface : Präsentationstechnik. Online verfügbar unter http://www.musicstore.de/de_DE/EUR/art-PCM0005090-000, zuletzt geprüft am 27.03.2015.

Fofana, Raby Florence (2015), 04.06.2015.

Greenberg, Ira (2007): Foundation Processing. [New York]: Ira Greenberg.

Greenberg, Ira; Xu, Dianna; Kumar, D. (2013): Processing. Creative coding and generative art in processing 2. Berkeley, Calif., London: Friends of Ed; Springer [distributor].

Halsall, Fred (1996): Data communications, computer networks, and open systems. 4th ed. Wokingham, England, Reading, Mass.: Addison-Wesley Pub. Co. (Electronic systems engineering series).

Home - GEjODOME Zelte - der Dome aus Holz. Online verfügbar unter http://www.gejodome.de/, zuletzt geprüft am 02.05.2015.

Kories, Ralf; Schmidt-Walter, Heinz (2010): Taschenbuch der Elektrotechnik. Grundlagen und Elektronik. 9., korrigierte Aufl. Frankfurt, M.: Deutsch.

Lambda Group : Innovative Solutions for Infrastructure | Telecom Industry | Defence | Satellite Industry. Online verfügbar unter http://lambdagroup.co.in/4_pair_stp_data_cable_shielded.php, zuletzt geprüft am 08.04.2015.

Lindner, Helmut; Brauer, Harry; Lehmann, Constans; Lindner, Harald (2004): Taschenbuch der Elektrotechnik und Elektronik. Mit 108 Tabellen. 8., neu bearb. Aufl. München [u.a.]: Fachbuchverl. Leipzig im Carl Hanser Verl.

mathe-lexikon.at. Online verfügbar unter http://www.mathe-lexikon.at/, zuletzt geprüft am 11.06.2015.

Musikinstrumente bei session. Online verfügbar unter http://www.session.de/, zuletzt geprüft am 11.05.2015.

Müller, Rolf (2006): Elektrotechnik. Lexikon für die Praxis. 2., stark bearb. und erw. Aufl. Berlin: Huss.

Nöding, Christian (2015): Was ist eigentlich DMX? Online verfügbar unter http://www.pcdimmer.de/dmx512-hardware/was-bedeutet-dmx, zuletzt aktualisiert am 31.03.2015, zuletzt geprüft am 31.03.2015.

professional, com: IT-News | Fachwissen für IT-Entscheider | com! professional. Online verfügbar unter http://www.com-magazin.de/, zuletzt geprüft am 08.04.2015.

Re-In Retail International GmbH: Elektronik, Technik, Werkzeug und mehr | voelkner - direkt günstiger. Re-In Retail International GmbH. Online verfügbar unter http://www.voelkner.de/, zuletzt geprüft am 11.05.2015.

Schenk, Joachim; Rigoll, Gerhard (2010): Mensch-Maschine-Kommunikation. Grundlagen von sprach- und bildbasierten Benutzerschnittstellen. Heidelberg [u.a.]: Springer.

Schiller, Brad (2010): The automated lighting programmer's handbook. 2nd ed. Oxford: Focal.

Schlessmann, Karl (2015), 08.07.2015.

Sierra, Kathy; Bates, Bert (2008): Java von Kopf bis Fuß. [behandelt Java 5.0 ; ein Buch zum Mitmachen und Verstehen]. 3., korr. Nachdr. Beijing [u.a.]: O'Reilly.

Simple Multi-Touch (SMT) Toolkit for Processing[Home]. Online verfügbar unter http://vialab.science.uoit.ca/smt/, zuletzt geprüft am 18.04.2015.

Thyristor Grundlagen. Online verfügbar unter http://elektronik-kurs.net/elektronik/thyristor-grundlagen/, zuletzt geprüft am 07.04.2015.

Unternehmen | tbm GmbH. Online verfügbar unter http://www.tbmgmbh.de/, zuletzt geprüft am 11.05.2015.

USITT. Online verfügbar unter http://www.usitt.org/content.asp?contentid=370, zuletzt geprüft am 30.03.2015.

US, Dell: Dell Official Site - The Power To Do More | Dell. Online verfügbar unter http://www.dell.com/, zuletzt geprüft am 11.05.2015.

Zühlke, Detlef (2012): Nutzergerechte Entwicklung von Mensch-Maschine-Systemen. Useware-Engineering für technische Systeme. 2., new bearb. Aufl. Heidelberg: Springer.

Anhang

Anhang A: Processing Code

Als Anhang A der Bachelorarbeit ist eine CD- Rom beigefügt. Darauf befindet sich der vollständige Processing Code in dem Format PDF und PDE. Wie die Inhalte wiedergegeben werden können, kann man der Datei „README“ entnehmen, welche sich ebenfalls auf der CD- Rom befindet.

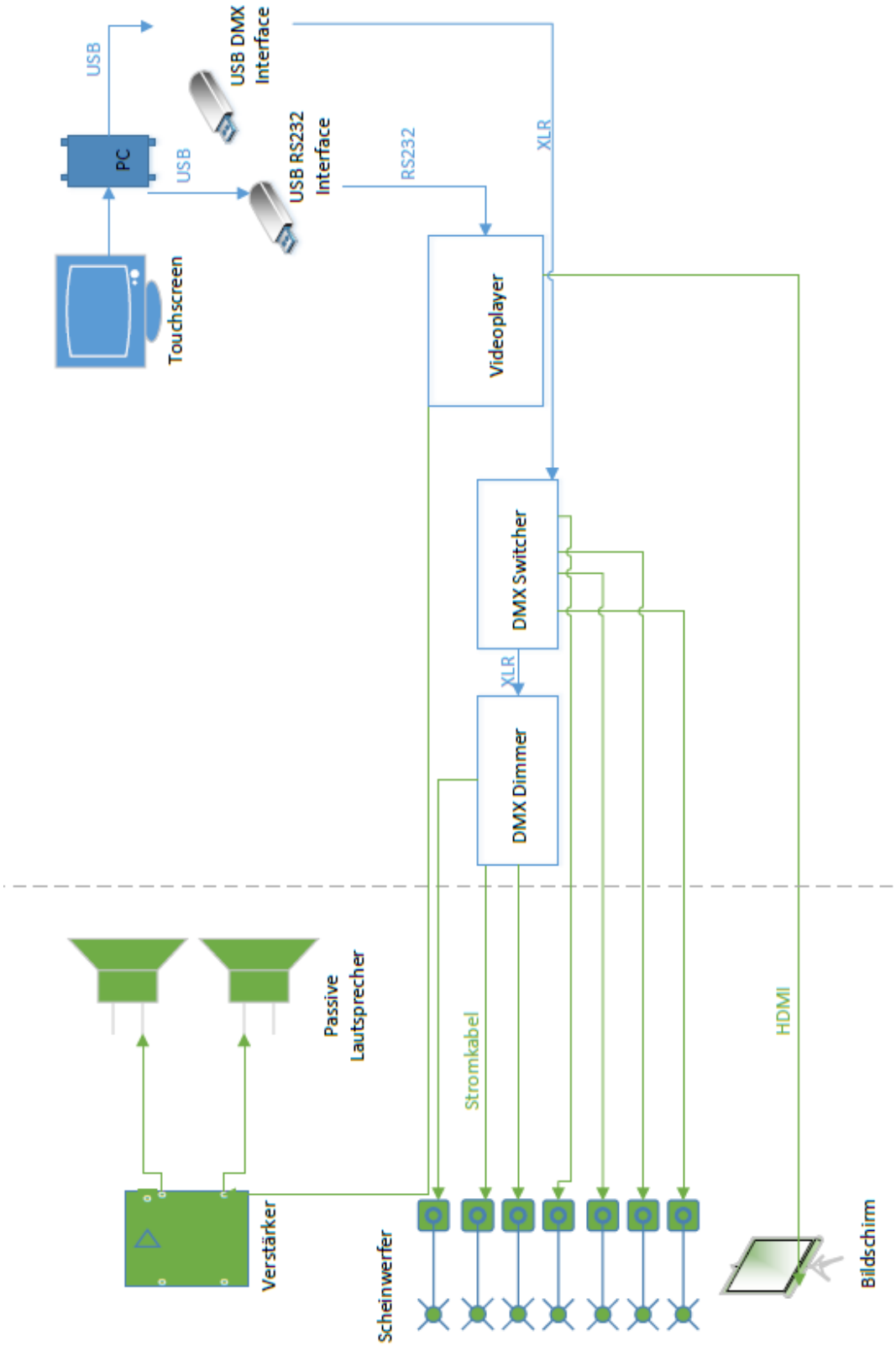
Anhang B: Geräteliste

Geräteliste Zeitmaschine

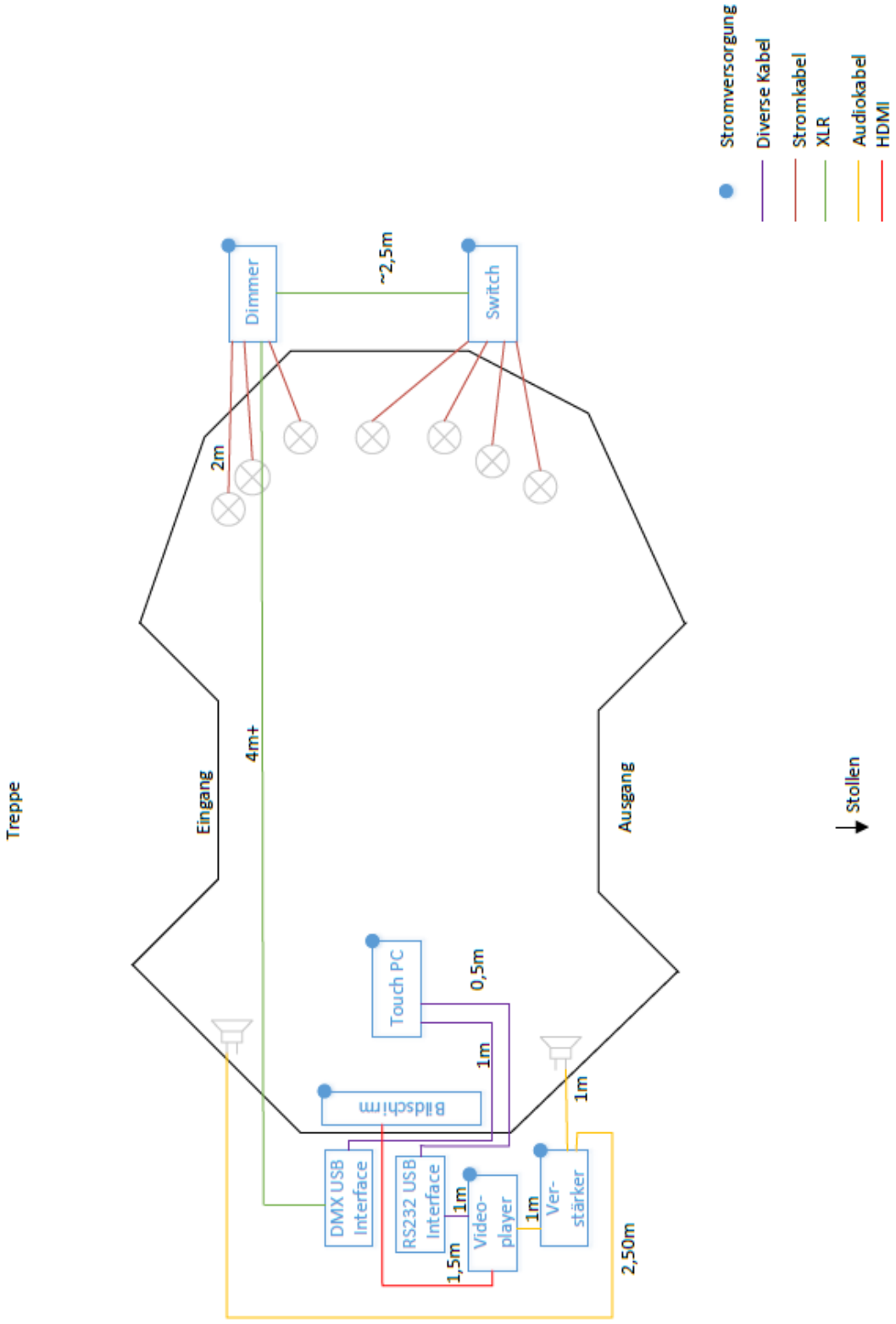
Anzahl	Gerät	Preis in € gesamt
1	**All in One PC mit Touchscreen** Dell Inspiron 237000	1200
1	**Videoplayer RS232** Videoplayer MPL031 tbm	390
1	**DMX Dimmerpack** EDX-4 Eurolite	100
1	**DMX Switch Pack** ERX-4 Eurolite	100
1	**USB DMX Dongle** Enttec DMX USB Mk2	150

1	**USB RS232 Dongle** Artikel-Nummer: A613681 Voelkner 	10
3	**XLR Kabel** 	24
1	**5 pin auf 3 pin DMX** ACCU-CABLE AC-DMXT/5M3F DMX Adapter 	4
1	**RS232 female/female** kenable 	5
		1983

Anhang C: technische Skizze



Anhang D: Positionierung und Kabelwege der Geräte



Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich diese Bachelorarbeit selbstständig verfasst und keine anderen, als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Zitate und Ausführungen, die anderen Werken wörtlich oder sinngemäß entnommen wurden, habe ich dahingehend kenntlich gemacht.

Außerdem wurde diese Arbeit in gleicher oder ähnlicher Fassung noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht.

Offenburg, den 16.07.2015

Marina Zajec